МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 3

з дисципліни “ Математичні та алгоритмічні основи комп’ютерної графіки”

Виконав

студент III курсу

групи КП-73

Булаєвський Ігор Олегович

(*прізвище, ім’я, по батькові*)

Варіант №3

Київ - 2020

Лістинг програми

PrintingImage.java

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import javafx.animation.*;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.shape.*;
import javafx.stage.Stage;
import javafx.util.Duration;

public class PrintingImage extends Application {
	private HeaderBitmapImage image; // приватне поле, яке зберігає об'єкт з інформацією про заголовок зображення
	private int numberOfPixels; // приватне поле для збереження кількості пікселів з чорним кольором
	private Stage primaryStage;
	private char[][] map;
	private int height, width;
	public PrintingImage() {
	}

	public PrintingImage(HeaderBitmapImage image) // перевизначений стандартний конструктор
	{
		this.image = image;
	}

	Group readAndPrintBmpFile(String path) throws Exception{
		Group root = new Group();
		ReadingImageFromFile.loadBitmapImage(path);
		this.image = ReadingImageFromFile.pr.image;
		this.width = (int) this.image.getWidth();
		this.height = (int) this.image.getHeight();
```

```
            int half = (int) image.getHalfOfWidth();

            Circle cir;
            int let = 0;
            int let1 = 0;
            int let2 = 0;
            this.map = new char[width][height];
// виконуємо зчитування даних про пікселі
            BufferedInputStream reader = new BufferedInputStream(new
                        FileInputStream("pixels.txt"));
            for (int i = 0; i < height; i++) // поки не кінець зображення по
висоті
            {
                  for (int j = 0; j < half; j++) // поки не кінець
зображення по довжині
                  {
                        let = reader.read(); // зчитуємо один символ з
файлу
                        let1 = let;
                        let2 = let;
                        let1 = let1 & (0xf0); // старший байт - перший
піксель
                        let1 = let1 >> 4; // зсув на 4 розряди
                        let2 = let2 & (0x0f); // молодший байт - другий
піксель
                        if (j * 2 < width) // так як 1 символ кодує 2
пікселі нам необхідно пройти до середини ширини зображення
                        {
                              cir = new Circle((j) * 2, (height - 1 - i), 1,

      Color.valueOf((returnPixelColor(let1)))); // за допомогою
стандартного
                              // примітива Коло радіусом в 1 піксель та
кольоромвизначеним за допомогою методу returnPixelColor малюємо
піксель
                              root.getChildren().add(cir); //додаємо об'єкт
в сцену
                              if (returnPixelColor(let1) == "BLACK") //
якщо колір пікселя чорний, то ставимо в масиві 1
                              {
                                    map[j * 2][height - 1 - i] = '1';
                                    numberOfPixels++; // збільшуємо
кількість чорних пікселів
```

```
                        } else {
                                map[j * 2][height - 1 - i] = '0';
                        }
                }
                if (j * 2 + 1 < width) // для другого пікселя
                {
                        cir = new Circle((j) * 2 + 1, (height - 1 -
                                        i), 1,
Color.valueOf((returnPixelColor(let2))));
                        root.getChildren().add(cir);
                        if (returnPixelColor(let2) == "BLACK") {
                                map[j * 2 + 1][height - 1 - i] = '1';
                                numberOfPixels++;
                        } else {
                                map[j * 2 + 1][height - 1 - i] = '0';
                        }
                }
        }
    }
    reader.close();
    return root;
}

int[][] writeMapToFile() throws Exception {
    int[][] black;
    black = new int[numberOfPixels][2];
    int lich = 0;
    BufferedOutputStream writer = new
BufferedOutputStream(new
                FileOutputStream("map.txt")); // записуємо
карту для руху по траекторії в файл
    for (int i = 0; i < height; i++) // поки не кінець зображення по
висоті
    {
        for (int j = 0; j < width; j++) // поки не кінець
зображення по  довжині
        {
            if (map[j][i] == '1') {
                black[lich][0] = j;
                black[lich][1] = i;
                lich++;
            }
            writer.write(map[j][i]);
```

```
                }
                writer.write(10);
            }
            writer.close();
            System.out.println("number of black color pixels = " +
numberOfPixels);
            return black;
        }

        double distance(int[] first, int[] second) {
            int dx = first[0] - second[0];
            int dy = first[1] - second[1];
            return Math.sqrt(dx * dx + dy * dy);
        }

        int findNearestPixel(int[] origin, List<int[]> pixels) {
            double minDistance = Double.MAX_VALUE;
            int minIndex = 0;
            for (int i = 0; i < pixels.size(); i++) {
                double curDistance = distance(origin, pixels.get(i));
                if (curDistance < minDistance) {
                    minDistance = curDistance;
                    minIndex = i;
                }
            }
            return minIndex;
        }

        Path createTransitionPath(int[][] black) {
            ArrayList<int[]> pixels = new ArrayList<>();
            for (int[] pixel: black) pixels.add(pixel);
            Path path = new Path();
            if (pixels.size() == 0) return path;

            int[] currentPixel = pixels.remove(0);
            path.getElements().add(new MoveTo(currentPixel[0],
currentPixel[1]));
            while (pixels.size() > 0) {
                int nextPixelIndex = findNearestPixel(currentPixel,
pixels);
                currentPixel = pixels.remove(nextPixelIndex);
                path.getElements().add(new LineTo(currentPixel[0],
currentPixel[1]));
```

```
        }
        return path;
}

Group drawSun() {
        Group group = new Group();
        Circle body = new Circle(0, 0, 40);
        body.setFill(Color.ORANGE);
        Circle leftEye = new Circle(-15, -5, 5, Color.BLACK);
        Circle leftEyeHelper = new Circle(-12, -1, 5, Color.ORANGE);
        Circle rightEye = new Circle(15, -5, 5, Color.BLACK);
        Circle rightEyeHelper = new Circle(12, -1, 5, Color.ORANGE);
        Line leftBrow = new Line(-25, -12, -15, -16);
        leftBrow.setFill(Color.BLACK);
        Line rightBrow = new Line(25, -12, 15, -16);
        rightBrow.setFill(Color.BLACK);
        Arc mouth = new Arc(0, 10, 20, 10,  10, -180);
        mouth.setFill(Color.RED);
        Polygon firstTooth = new Polygon(5, 15,
                        -5, 16,
                        -5, 11,
                        5, 10
        );
        firstTooth.setFill(Color.WHITE);

        group.getChildren().addAll(body,
                        leftEye, leftEyeHelper, rightEye, rightEyeHelper,
                        leftBrow, rightBrow, mouth, firstTooth,
                        new Circle(-12, -1, 1, Color.BLACK),
                        new Circle(-9, 0, 1, Color.BLACK),
                        new Circle(-15, 1, 1, Color.BLACK),
                        new Circle(-13, 3, 1, Color.BLACK),
                        new Circle(-18, 4, 1, Color.BLACK),
                        new Circle(12, -1, 1, Color.BLACK),
                        new Circle(9, 0, 1, Color.BLACK),
                        new Circle(15, 1, 1, Color.BLACK),
                        new Circle(13, 3, 1, Color.BLACK),
                        new Circle(18, 4, 1, Color.BLACK));

        return group;
}

Group drawRays() {
```

```
        Group group = new Group();
        Line[] rays = new Line[] {
                new Line(0, 50, 0, 80),
                new Line(0, -50, 0, -80),
                new Line(50, 0, 80, 0),
                new Line(-50, 0, -80, 0),
                new Line(-25, 43, -40, 70),
                new Line(25, -43, 40, -70),
                new Line(-43, 25, -68, 40),
                new Line(43, -25, 68, -40),
                new Line(25, 43, 40, 70),
                new Line(-25, -43, -40, -70),
                new Line(25, 43, 40, 70),
                new Line(43, 25, 70, 40),
                new Line(-43, -25, -70, -40)
        };
        for (Line line: rays) {
            line.setStrokeWidth(5);
            line.setStroke(Color.ORANGE);
        }
        group.getChildren().addAll(rays);
        return group;
    }

    void drawMainScene(Group root) {
        Scene scene = new Scene(root, width, height);
        scene.setFill(Color.BLACK);
        primaryStage.setScene(scene); // ініціалізуємо сцену
        primaryStage.show(); // візуалізуємо сцену
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        this.primaryStage = primaryStage;
        Group root =
readAndPrintBmpFile("sources\\my_trajectory.bmp");
        drawMainScene(root);
        int[][] black = writeMapToFile();
        Path transitionPath = createTransitionPath(black);

        Group body = drawSun();
        Group rays = drawRays();
        root.getChildren().addAll(body, rays);
```

```
        PathTransition pathTransitionBody = new PathTransition();
        pathTransitionBody.setPath(transitionPath);
        pathTransitionBody.setDuration(Duration.seconds(30));
        pathTransitionBody.setNode(body);

    pathTransitionBody.setCycleCount(PathTransition.INDEFINITE);

        PathTransition pathTransitionRays = new PathTransition();
        pathTransitionRays.setPath(transitionPath);
        pathTransitionRays.setDuration(Duration.seconds(30));
        pathTransitionRays.setNode(rays);

    pathTransitionRays.setCycleCount(PathTransition.INDEFINITE);


        RotateTransition rotTransitionRays = new RotateTransition();
        rotTransitionRays.setFromAngle(0);
        rotTransitionRays.setToAngle(360);
        rotTransitionRays.setDuration(Duration.seconds(10));
        rotTransitionRays.setNode(rays);

    rotTransitionRays.setCycleCount(RotateTransition.INDEFINITE);

        ScaleTransition scaleTransitionBody = new ScaleTransition();
        scaleTransitionBody.setFromX(1);
        scaleTransitionBody.setFromY(1);
        scaleTransitionBody.setToX(0.5);
        scaleTransitionBody.setToY(0.5);
        scaleTransitionBody.setDuration(Duration.seconds(5));
        scaleTransitionBody.setAutoReverse(true);
        scaleTransitionBody.setNode(body);

    scaleTransitionBody.setCycleCount(ScaleTransition.INDEFINITE);

        List<StrokeTransition> strokeTransitions = new
ArrayList<>();
        for (Node child: rays.getChildren()) {
            if (child instanceof Line) {
                StrokeTransition transition = new
StrokeTransition(Duration.seconds(5), (Line)child);

    transition.setCycleCount(FillTransition.INDEFINITE);
```

```
                    transition.setAutoReverse(true);
                    transition.setFromValue(Color.ORANGE);
                    transition.setToValue(Color.RED);
                    strokeTransitions.add(transition);
                }
            }

            ParallelTransition parTransition = new ParallelTransition();
            parTransition.getChildren().addAll(pathTransitionBody,
pathTransitionRays,
                    rotTransitionRays, scaleTransitionBody);
            parTransition.getChildren().addAll(strokeTransitions);
                    parTransition.play();
        }

        private String returnPixelColor(int color) // метод для
співставлення кольорів 16-бітного зображення
        {
            String col = "BLACK";
            switch (color) {
                case 0:
                    return "BLACK"; //BLACK;
                case 1:
                    return "LIGHTCORAL"; //LIGHTCORAL;
                case 2:
                    return "GREEN"; //GREEN
                case 3:
                    return "BROWN"; //BROWN
                case 4:
                    return "BLUE"; //BLUE;
                case 5:
                    return "MAGENTA"; //MAGENTA;
                case 6:
                    return "CYAN"; //CYAN;
                case 7:
                    return "LIGHTGRAY"; //LIGHTGRAY;
                case 8:
                    return "DARKGRAY"; //DARKGRAY;
                case 9:
                    return "RED"; //RED;
                case 10:
                    return "LIGHTGREEN";//LIGHTGREEN
                case 11:
```

```
                        return "YELLOW"; //YELLOW;
                    case 12:
                        return "LIGHTBLUE"; //LIGHTBLUE;
                    case 13:
                        return "LIGHTPINK"; //LIGHTMAGENTA
                    case 14:
                        return "LIGHTCYAN"; //LIGHTCYAN;
                    case 15:
                        return "WHITE"; //WHITE;
                }
                return col;
        }

        public static void main(String args[]) {
                launch(args);
        }
}
```

Скріншоти результатів